INS FORAT DEL VENT– INTERNATIONAL BACCALUREATE

INFORMÀTICA I - POO

TEMA 7: ARRAYLISTS (COL·LECCIONS)

En moltes ocasions els nostres programes han de treballar amb una gran i indefinida quantitat de dades. En aquests casos podem treballar amb arrays, però els arrays tenen la seva llargària fixa. Per superar aquesta limitació apareixen les **Col·leccions**, que són classes predefinidas que permeten emmagatzemar dades del mateix tipus. Les Col·leccions tenen les següents característiques:

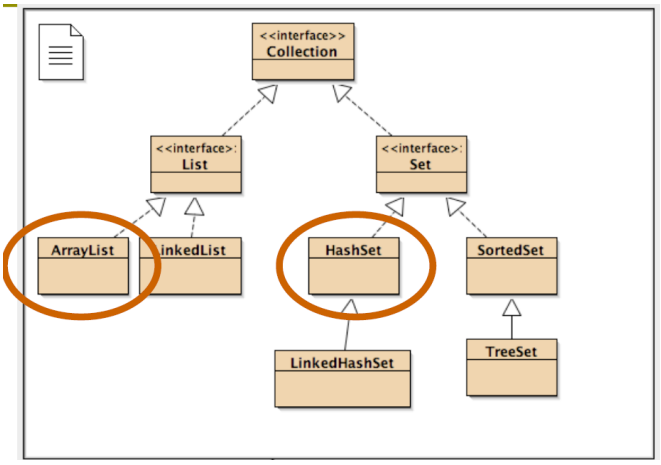
- La seva mida pot variar dinàmicament (una col·lecció pot créixer i reduir-se mentre s'executa el programa, per la qual cosa presenta un ús més eficient de la memòria).
- Es presenten de diverses formes o estructures: llistes, conjunts, mapes.
- Només treballen amb objectes (Hi ha classes envoltori ("*wrapper*") per a tipus primitius)

Java ofereix diverses classes de col·leccions, com ara List, LinkedList, Stack, TreeSet, HashSet i Vector. En el cas concret de les **llistes**, són estructures de dades que presenten les següents avantatges:

- Una llista pot créixer o disminuir segons sigui necessari.
- En una llista es respecta l'ordre d'inserció dels elements i pot haver elements duplicats.
- Les dades d'una llista també estan indexats, però mitjançant *iteradors*
- Les llistes ofereixen mètodes específics per operacions comuns com inserir i esborrar elements.
- Les llistes son una ***classe genèrica*** (pot contenir qualsevol tipus de dades i objectes), però s'ha d'indicar el tipus concret d'objecte mitjançant: **<...> .** Per exemple:

  List<E> // Llista d'elements de tipus E (qualsevol classe)

Entre d'altres, Java ofereix la **classe ArrayList**, que es troba en el paquet **java.util.ArrayList**



Podem veure que els **ArrayList** és un tipus de **List**, es diu que és una implementació de List (noteu que List és un interface, es a dir una classe sense implementar).

- **Declaració d'un ArrayList**

Per poder usar un ArrayList, durant la declaració s'indicarà el tipus dels elements dins dels: < > com un tipus de "paràmetre". El tipus ha de ser una classe, no es poden fer servir tipus de dades primitives (int, double…). Per exemple farem:

ArrayList de Strings: ***ArrayList<String> notes = new ArrayList<String>();***

ArrayList com a List de Strings: ***List<String> notes = new ArrayList<String>();***

- **Mètodes útils d'un ArrayList**

– ***add***: afegeix un element
boolean add(E element) : l'element de tipus E s'afegeix al final de la llista
void add(int posicio, E element): l'element de tipus E s'insereix a la posició
Per exemple: *notes.add(nota);*

– ***get***: retorna l'element situat a la posició indicada
E get(int posicio)
Per exemple: *nota = notes.get(n);*

– ***remove***: elimina l'element situat a la posició indicada
E remove(int posicio)
boolean remove(E element)
Per exemple: *notes.remove(n);*

– ***set***: canvia un element
E set(int posicio, E element)
Per exemple: *notes.set(n,nota);*

– ***size***: longitud de l'arraylist
int size()
Per exemple*: notes.size();*

– **altres**: buidar, si esta buit, i si conté un element
void clear()
boolean contains(E element)
boolean isEmpty()

Mètodes ArrayList	Descripció
ArrayList <E> ()	Constructor per crear un Arralist buit
boolean add(E obj)	L'element de tipus E s'afegeix al final de la llista. Retorna true.
void add(int posicio, E obj)	L'element de tipus E s'insereix abans de l'element a la posició o després de l'últim element si l'índex és igual a la mida de la llista.
E get(int posicio)	Retorna l'element situat a la posició indicada
E remove(int posicio)	Elimina i retorna l'element situat a la posició indicada
boolean remove(E obj)	Elimina l'element E i retorna true.
E set(int posicio, E obj)	Substitueix l'element a l'índex per obj i retorna l'element antic.
int size()	Retorna el nombre d'elements actual a la llista.
boolean isEmpty()	Retorna true si la llista és buida i false en cas contrari.
boolean contains(E obj)	Retorna true si a la llista hi ha l'element E
void clear()	Deixa la llista sense elements

- **Recorregut de col·leccions: for-each i iteradors**

Hi ha una estructura especial per recórrer col·leccions, el **for-each:**

```
for( Tipus element: col·lecció) {
        ...
}
```

Per exemple:
```
for( String nota: notes) {
        System.out.println(nota);.
}
```

Sempre podem fer el recorregut clàssic, ja sigui amb un for:

```
for(int i = 0; i < notes.size(); i++) {
        System.out.println(notes.get(i));
}
```

o també amb un while:
```
int i = 0;
while(i < notes.size()) {
        System.out.println(notes.get(i));
}
```

També es poden utilitzar **iteradors:**
```
Iterator<String> it = notes.iterator();
while (it.hasNext()) {
        String nota = it.next();
        System.out.println(nota);
}
```

Un objecte **Iterator<E>** permet l'accés seqüencial als elements d'una col·lecció i realitzar recorreguts sobre aquesta. Ve a ser com un *cursor* que assenyala un element de la col·lecció. Pel procés d'iteració s'han d'utilitzar els mètodes propis de la classe (interface) **Iterator<E>**. Estan definits segons:

```
public interface Iterator<E> {
        boolean hasNext();// Comprova si hi ha següent element
        E next();           // Retorna el següent element i mou l'iterador
        void remove();      // S'invoca després de next() per a eliminar un element
}
```

El mètode *iterator()* retorna un iterador inicialitzat en el primer element d'una col·lecció:

```
Iterator <E > it = list.iterator();
```

El mètode *hasNext()* retorna true mentre hi hagi algun element pendent de recórrer:

```
while ( it.hasNext () ) {
        System.out.println(it.next());
}
```

El mètode *next()* retorna l'element al qual es pot accedir des de el iterador (cursor) i deixa aquest en la següent posició (post-increment), tal com també es fa en el recorregut d'un array. A tenir en compte que:

- Si no hi ha següent element, *next()* llança una excepció *NoSuchElementException*.
- El mètode *remove()* permet eliminar elements de la col·lecció. És l'única manera per a eliminar elements durant el recorregut, si no es llança una excepció *ConcurrentModificationException*.
- Només es pot fer una invocació a *remove()* per a cada invocació a *next()*. Si no, es llançarà una excepció *IllegalStateException*.
- Si es modifica la col·lecció, tots els iteradors quedaran invalidats, excepte en el cas que s'hagi aplicat el mètode *remove()*.
- No es pot modificar una col·lecció dins d'un bucle *for-each*.

- **Còpia d'un ArrayList en un altre**

Un ArrayList manté una referència, com els arrays, a la memòria. Si copiem aquesta referencia podem fer una còpia de l'ArrayList original. Llavors, per fer una còpia d'un ArraList a un altre, passarem la referència de l'ArrayList original al constructor del nou:

```
ArrayList<String> original = new ArrayList<String>();
//copia compartirà el mateix contingut que original
ArrayList<String> copia = original;
//copia2 serà una copia independent d'original
ArrayList<String> copia2= new ArrayList<String>(original);
```

- **Funcions amb ArrayLists com a paràmetres**

De igual manera que els arrays, un ArrayList es pot fer servir com a paràmetre o valor de retorn d'una funció.

Exemple: funció que rep un ArrayList y retorna la llista invertida

```
public static ArrayList<String> reverse(ArrayList<String> names) {
        // Crea un arraylist per la llista resultat
        ArrayList<String> result = new ArrayList<String>();
        // Fa el recorregut de la llista de noms en ordre invers (últim a primer)
        for (int i = names.size() - 1; i >= 0; i--) {
                // Afegeix cada nom a l'arraylist result
                result.add(names.get(i));
        }
        return result;
}
```

- **Wrappers i autoboxing**

A diferència d'un array, un ArrayList només pot contenir objectes, en cap cas tipus de dades primitius. Com que els números no són objectes en Java, no els podrem inserir directament a un ArrayList, ja que no hi ha un ArrayList<int> o un ArrayList<double>.

Per emmagatzemar seqüències de números en un ArrayList, els haurem de convertir en objectes mitjançant classes d'embolcall. Java ofereix classes d'embolcall ("**wrapper**") per als tipus primitius. Les principals classes Wrapper són:

Tipus primitiu	**Classe Wrapper**
booelan	Booelan
char	Character
double	Double
int	Integer

A l'hora de fer aquest tipus de conversió, hem de tenir en compte que:

– Les conversions són automàtiques usant "*auto-boxing*" (la conversió entre els tipus primitius i les classes d'embolcall corresponents és automàtica. Aquest procés s'anomena auto-boxing).
– Es passa de tipus primitiu a objecte amb les classes "*Wrapper*" (embolcall)

Per exemple:
```
double x = 29.95;
Double num;          //Double és la classe Wrapper
num = x;             // boxing (num és un objecte)
```

Per exemple:
```
double x;
Double num = 29.95;
x = num;             // unboxing
```

Llavors, no es poden usar tipus primitius en un ArrayList, però si les seves classes wrapper. Per exemple, podem declarar un ArrayList amb classes Wrapper pel tipus primitiu double:

```
double x = 19.95;
ArrayList<Double> data = new ArrayList<Double>();
data.add(29.95);       // boxing
data.add(x);           // boxing
double x = data.get(0); // unboxing
```

- **Diferència entre arrays i ArrayLists**

Per treballar amb un array: – farem ús de índexs ([i])
– farem ús del mètode *data.length*

Per exemple, per determinar el màxim d'un array:

```
double largest = data[0];
for (int i = 1; i < data.length; i++) {
        if (data[i] > largest) {
                largest = data[i];
        }
}
```

Per treballar amb un ArrayList: – farem ús del mètode *get(i)*
– farem ús del mètode *data.size()*

Per exemple, per determinar el màxim d'un ArrayList:

```
double largest = data.get(0);
for (int i = 1; i < data.size(); i++) {
        if (data.get(i) > largest) {
                largest = data.get(i);
        }
}
```

En general, es recomanable l'ús d'arrays si:
– la mida de l'array no canvia
– tenim una seqüència gran de tipus primitiu
i es recomanable l'ús d'un ArrayList:
– en els casos no contemplats anteriorment
– especialment si tenim un nombre desconegut de dades
Recordeu que per saber la mida d'un:

- Array → a.length
- Arraylist → a.size()
- String → a.length()

BASE PRÀCTICA

- **Programes d'exemple**

1.- Comproveu la sortida del següent programa que treballa amb un ArrayList:

```
import java.util.*;   // per treballar amb ArrayList

public class ArrayListDemo {
   public static void main(String[] args) {
      // Instancia d'un objecte ArrayList de Strings
      ArrayList<String> list = new ArrayList<>();
      // Afegim items a l'Arraylist
      list.add("Barcelona");
      list.add("Paris");
      list.add("Berlin");
      list.add("Londres");
      // Els treiem per pantalla amb diversos mètodes
      System.out.println(list.get(0));
      System.out.println(list.get(1));
      System.out.println(list.get(2));
      System.out.println(list.get(3));
      for( String ciutat: list) {
          System.out.println(list);.
      }
      for(int i = 0; i < list.size(); i++) {
          System.out.println(list.get(i));
      }
      Iterator<String> it = list.iterator();
      while (it.hasNext()) {
         String ciutat = it.next();
         System.out.println(ciutat);
      }
   }
}
```

2.- Comproveu la sortida del següent programa que treballa amb ArrayLists:

```
import  java.util.ArrayList;
import  java.util.Iterator;

class Arraylists {
   public static void main(String[] args) {
      //Creant un ArrayList de Strings
      ArrayList<String> llista = new ArrayList<String>();
      //Mida inicial
      System.out.println("Mida del ArrayList quan es crea:" + llista.size());
      //Afegint elements
      llista.add("Joan");
      llista.add("Pep");
      llista.add("Josep");
      System.out.println("Elements a l'ArrayList:");
      System.out.println(llista);
      //Afegint element a la posició 2:
      llista.add(2, "Alfons");
      System.out.println("Nova mida del ArrayList:" + llista.size());
      System.out.println("Elements a l'ArrayList: " + llista);
      //Esbborrem element concret
      llista.remove("Pep");
      System.out.println("Nova mida del ArrayList:" + llista.size());
      System.out.println("Elements a l'ArrayList: " + llista);
      //Esbborrem element per index
      llista.remove(1);
      System.out.println("Nova mida del ArrayList:" + llista.size());
      System.out.println("Elements a l'ArrayList: " + llista);
      //Recorregut de la llista
      for(int i = 0; i < llista.size();i++) {
         System.out.println(llista.get(i));
      }
      System.out.println("Recorregut amb iteradors");
      Iterator<String> it = llista.iterator();
      while(it.hasNext()) {
         System.out.println(it.next());
      }
      // Verifica si la lista conte Alfons
      System.out.println(llista.contains("Alfons"));
      //ArrayList de enters
      ArrayList<Integer> numbers = new ArrayList<Integer>();
      numbers.add(4);
      numbers.add(8);
      numbers.add(2);
      System.out.println("Elements de la llista: ");
      System.out.println(numbers);
      numbers.set(1, 6);       //Posem el numero 6 a la posició 1
      System.out.println("Elements de la llista: ");
      System.out.println(numbers);
      //Recorregut de la llista
      for(Integer n: numbers) {
         System.out.println(n);
      }
      numbers.remove(2);       //Esborrem element posició 2
      System.out.println("Elements de la llista: ");
      System.out.println(numbers);
      //Esborrem la llista numbers
      numbers.clear();
      if(numbers.isEmpty()) System.out.println("Elements de la llista : " + numbers);
```

```
        //Creem una altre Arraylist de Strings
        ArrayList<String> original = new ArrayList<String>();
        original.add("Hola");
        //Fem una copia
        ArrayList<String> copia = original;
        copia.add("Adeu");
        //Fem una segona copia
        ArrayList<String> copia2= new ArrayList<String>(original);
        System.out.println("Elements a l'ArrayList original: " + original);
        System.out.println("Elements a l'ArrayList copia: " + copia);
        System.out.println("Elements a l'ArrayList copia2: " + copia2);
    }
}
```

3.- Feu una llista amb el primers 9 números, i mostreu per pantalla només els números parells:

```
import java.util.ArrayList;
import java.util.Iterator;

class Arraylists {
    public static void main(String[] args) {

            ArrayList<Integer> list = new ArrayList<>();

            list.add(9);
            list.add(3);
            list.add(5);
            list.add(2);
            list.add(6);
            list.add(4);
            list.add(8);
            list.add(1);

            // llista total
            for(Integer n : list) {
              System.out.print(n + " ");
            }
            System.out.println();

            // esborrem números parells
            for(int i=0; i < list.size(); i++) {
              if(list.get(i) % 2 != 0) {
                list.remove(i);
                i--;      //hem de fer això ja que si esborrem, l'index s'avança una posició
              }
            }

            // llista
            for(Integer n : list) {
              System.out.print(n + " ");
            }
            System.out.println();
        }
}
```

4.- Feu una classe anomenada Coche amb els atributs: marca, model i preu, i els seus setters i getters i constructors. Després feu una altra classe Principal, que mitjançat una Arraylist de Coches demani les dades dels cotxes d'una botiga, inicialment considerem tres cotxes, i mostri per pantalla un llistat complet dels tres cotxes a partir d'un mètode anomenat *mostrarCotxesBotiga*

```
public class Coche {
    private String marca;
    private String model;
    private double preu;

    public Coche(String marca, String model, double preu) {
        this.marca = marca;
        this.model = model;
        this.preu = preu;
    }

    public Coche() {
    }

    public String getMarca() {
        return marca;
    }

    public void setMarca(String marca) {
        this.marca = marca;
    }

    public String getModel() {
        return model;
    }

    public void setModel(String model) {
        this.model = model;
    }

    public double getPreu() {
        return preu;
    }

    public void setPreu(double preu) {
        this.preu = preu;
    }

}


public class Main {

    public static void main(String[] args) {

        int num = 3;
        ArrayList<Coche> botiga = new ArrayList<>();

        for (int i = 0; i < num; i++) {
            Scanner teclat = new Scanner (System.in);
            System.out.println("Dades cotxe num "+ (i+1) +" : ");
            System.out.println("Marca: ");
            String marca = teclat.next();
```

```
            System.out.println("Model: ");
            String model = teclat.next();
            System.out.println("Preu: ");
            double preu = teclat.nextDouble();
            System.out.println("\n\n");

            Coche aux = new Coche(marca,model,preu);
            botiga.add(aux);
        }

        mostrarCotxesBotiga(botiga);

    }

    public static void mostrarCotxesBotiga(ArrayList<Coche> botiga) {
        for (int i = 0; i < botiga.size(); i++) {
            System.out.println("Cotxe "+(i+1)+" --> MARCA: "+botiga.get(i).getMarca()+" MODEL:
"+botiga.get(i).getModel()+" PREU:"+botiga.get(i).getPreu());
        }
    }
}
```

- **Programes per desenvolupar**

1.- Completeu el programa ArrayList1 implementant els dos mètodes: Method1 i Method2.

Mètode 1: creeu una ArrayList de valors enters, afegiu els números del 10 al 100 comptant de 10 en 10 a la llista i, a continuació, recorreu la llista dues vegades imprimint-ne el contingut. La primera vegada imprimeix amb un bucle for normal i la segona vegada amb un bucle for millorat.

Mètode 2: creeu una ArrayList de paraules, afegiu paraules a la llista fins que s'escrigui el delimitador "fi" i, a continuació, recorreu la llista dues vegades imprimint el seu contingut. La primera vegada imprimeix amb un bucle for normal i la segona vegada amb un bucle for millorat. No afegiu la paraula "fi" a la llista.

Copieu el codi següent en un fitxer font anomenat ArrayList1.java:

```
import java.util.*;      // needed for ArrayList

public class ArrayListPractice
{
   static Scanner keyboard = new Scanner(System.in);

   public static void main(String[] args)
   {
      System.out.println("Metode 1");
      System.out.println("--------");
      System.out.println();
      Method1();
      System.out.println();
      System.out.println("Metode 2");
```

```
    System.out.println("--------");
    System.out.println();
    Method2();
  }

  public static void Method1()
  {

  }

  public static void Method2()
  {

  }
}
```

2.- Volem implementar un programa que permeti a un usuari introduir informació de contacte a una llista de telèfons. La informació de contacte inclou el nom i el número de telèfon d'una persona. El problema s'ha descompost en les dues classes següents: *Contacte* i *GuiaTelefonica*. La classe *Contacte* emmagatzema informació sobre cada contacte. Copieu el codi següent en un fitxer anomenat Contacte.java:

```
public class Contacte {
  private String name;
  private String phone;

  public Contacte(String n, String p) {
    name = n;
    phone = p;
  }
  public String getName() {
    return name;
  }
  public String getPhone() {
    return phone;
  }
  @Override
  public String toString() {
    return name + " -----> " + phone;
  }
}
```

La classe *GuiaTelefonica* és un client, té relació, de la classe *Contact*. Utilitza l'agregació (relació "té una"). La classe *GuiTelefonica* gestiona el programa permetent que un usuari introdueixi informació de contacte al programa. Un cop completada l'entrada de dades, la classe mostra la informació de contacte introduïda. La classe es pot subdividir en dos mètodes: getContacteInfo i printContacteInfo. El mètode getContacteInfo és responsable de permetre a un usuari introduir informació dels seus contactes, en un nombre indeterminat. El mètode printContacteInfo és responsable de

mostrar la informació de tots els contactes emmagatzemats a l'ArrayList. Copieu el codi següent en un fitxer font anomenat GuiaTelefoncia.java:

```
import java.util.*;
public class GuiaTelefonica {
  private ArrayList<Contacte> phoneList = new ArrayList<Contacte>();

  public void getContacteInfo() {...}

  public void printContacteInfo() {...}

  public static void main(String[] args) {
    GuiaTelefonica app = new GuiaTelefonica();
    app.getContactInfo();
    app.printContactInfo();
  }
}
```

La sortida de programa ha de seguir l'exemple:

Introdueix el nom i cognom de l'usuari: Pep Gomez
Introdueix el telèfon de l'usuari: 123456789
Vols afegir un altre [Y o N]: Y
Introdueix el nom i cognom de l'usuari: Joan Salas
Introdueix el telèfon de l'usuari: 111222333
Vols afegir un altre [Y o N]: Y
Introdueix el nom i cognom de l'usuari: Arnau Mas
Introdueix el telèfon de l'usuari: 987789987
Vols afegir un altre [Y o N]: N

-------- GUIA TELEFONICA --------
Pep Gomez -----> 123456789
Joan Salas -----> 111222333
Arnau Mas -----> 987789987

- **Programa opcional**

1.- Implementar un programa que demani a l'usuari el nombre d'alumnes d'un grup/aula i a partir de les notes de cadascú d'ells i el seu DNI calculi la mitjana i ens mostri una llista dels aprovats i dels suspesos, i que també ens mostri una llista ordenada per ordre creixent de la nota de tot el grup d'alumnes. També s'ha de tenir la possibilitat de buscar els alumnes que hi ha amb una determinada nota. Heu d'utilitzar les dues classes: Alumne i Aula. Feu el diagrama UML corresponent a la vostra proposta.